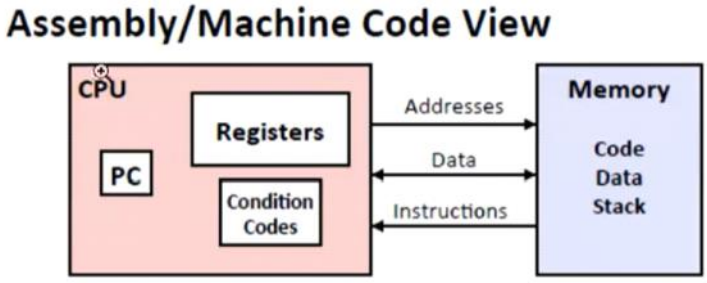
Machine-Level Programming

2018年6月23日 星期六 上午11:47

Instructure Set Architecture: the contract between software and hardware, where instructions lie(eg. mov)
Microarchitecture: ISA design and organization, include execution units, pipelining, cache architecture, control units
Machine Code: excutable binary instruction
Assembly Code: The text form of Machine Code (eg. mov %rax %rbx)



Registers: memory location given by name (eg. Rax). Small, fast storage locations directly within the CPU
Memory: array of addresses. All addresses store in it

```
*dest = t;
```

- **C Code**
  - Store value **t** where designated by **dest**

```
movq %rax, (%rbx)
```

- **Assembly**
  - Move 8-byte value to memory
    - Quad words in x86-64 parlance
  - Operands:
    - **t:** Register **%rax**
    - **dest:** Register **%rbx**
    - ***dest:** Memory **M[%rbx]**

```
0x40059e:  48 89 03
```

- **Object Code**
  - 3-byte instruction
  - Stored at address **0x40059**

Moving Data

- Moving Data

  `movq` ***Source, Dest***

- Operand Types
  - ***Immediate:*** Constant integer data
    - Example: `$0x400, $-533`
    - Like C constant, but prefixed with '$'
  - ***Register:*** One of 16 integer registers
    - Example: `%rax, %r13`
    - But `%rsp` reserved for special use
    - Others have special uses for particular instructions (later on that)
  - ***Memory:*** 8 consecutive bytes of memory at address given by register
    - Simplest example: `(%rax)`
    - We will see various other "address modes" later.

movq Operand Combinations

	Source	Dest	Src,Dest	C Analog
movl	Imm	Reg	`movq $0x4,%rax`	`temp = 0x4;`
		Mem	`movq $-147,(%rax)`	`*p = -147;`
	Reg	Reg	`movq %rax,%rdx`	`temp2 = temp1;`
		Mem	`movq %rax,(%rdx)`	`*p = temp;`
	Mem	Reg	`movq (%rax),%rdx`	`temp = *p;`

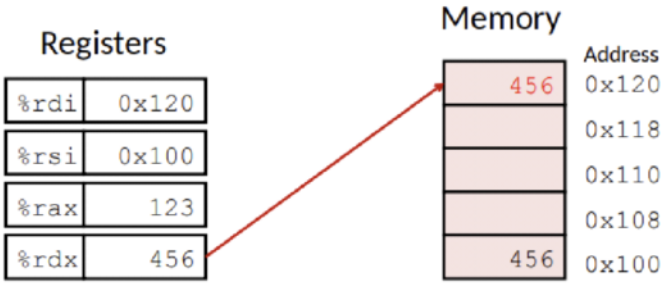
No memory-to-memory instruction

Example of Simple Addressing Modes

```
void swap (long *xp, long *yp)
{
  long t0 = *xp;
  long t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

```
swap:
    … some setup code
    movq    (%rdi), %rax
    movq    (%rsi), %rdx
    movq    %rdx, (%rdi)
    movq    %rax, (%rsi)
   … wrap-up code
    ret
```

Understanding Swap()




```
swap:
    movq    (%rdi), %rax  # t0 = *xp
    movq    (%rsi), %rdx  # t1 = *yp
    movq    %rdx, (%rdi)  # *xp = t1
    movq    %rax, (%rsi)  # *yp = t0
    ret
```

Complete Memory Addressing Modes

- **Most General Form**

  D(Rb,Ri,S) **Mem[Reg[Rb]+S*Reg[Ri]+ D]**

  - D: Constant "displacement" 1, 2, or 4 bytes
  - Rb: Base register: Any of 16 integer registers
  - Ri: Index register: Any, except for `%rsp`
  - S: Scale: 1, 2, 4, or 8 (*why these numbers?*)

- **Special Cases**

  (Rb,Ri) **Mem[Reg[Rb]+Reg[Ri]]**

  D(Rb,Ri) **Mem[Reg[Rb]+Reg[Ri]+D]**

  (Rb,Ri,S) **Mem[Reg[Rb]+S*Reg[Ri]]**

movq: copy the data from one place, then move it to another.
leaq: put the effective address of the operand into the destination register
Calculate the address without actually access the memory location